АЛГОРИТМ РЕПРЕЗЕНТАТИВНОГО СЭМПЛИНГА ДЛЯ ПАРАЛЛЕЛЬНЫХ РЕЛЯЦИОННЫХ СИСТЕМ БАЗ ДАННЫХ

Д.Д. Янцен, М.Л. Цымблер

Южно-Уральский государственный университет (Челябинск)

Введение. В настоящее время *сэмплинг* применяется в широком спектре приложений, связанных с обработкой сверхбольших баз данных: интеллектуальный анализ данных [10], построение гистограмм [17], тестирование программного обеспечения для генерации тестовых данных [25], приблизительная оценка стоимости исполнения запросов [5] и др. Сэмпл представляет собой часть оригинальной базы данных, имеющую меньший размер, что позволяет сократить время обработки базы данных за счет возможной потери точности результатов обработки. *Случайный сэмплинг* предполагает отбор кортежей базы данных без учета значений их атрибутов, что снижает точность и повторяемость результатов обработки. При *репрезентативном сэмплинге* отбор кортежей осуществляется с сохранением важных особенностей оригинальной базы данных: относительное количество кортежей, связанных посредством внешних ключей, относительное количество значений атрибута и др. [7].

На сегодня научное сообщество признает *параллельные системы баз данных* [3] как практически единственное эффективное средство для организации обработки сверхбольших баз данных. Базисной концепцией параллельных систем баз данных является *фрагментный параллелизм*, предполагающий разбиение отношений базы данных на горизонтальные фрагменты, которые могут обрабатываться независимо на разных узлах кластерной вычислительной системы. Однако существующие методы репрезентативного сэмплинга не приспособлены для систем баз данных на основе фрагментного параллелизма, поскольку не учитывают особенности распределения фрагментов отношений базы данных по вычислительным узлам кластерной системы.

В настоящей работе предлагается алгоритм репрезентативного сэмплинга реляционной базы данных для параллельных систем баз данных. Статья организована следующим образом. В разделе 1 представлен краткий обзор методов сэмплинга реляционных баз данных. Раздел 2 содержит описание предлагаемого алгоритма. В разделе 3 рассмотрены результаты вычислительных экспериментов, в которых исследуются свойства предложенного алгоритма. В заключении суммируются полученные результаты и рассматриваются направления дальнейших исследований в данной области.

1. Обзор работ в области сэмплинга данных. В настоящее время существует большое количество подходов к сэмплингу баз данных, которые, в основном, ориентированы на конкретные приложения: тестирование программного обеспечения, интеллектуальный анализ данных, нахождение приблизительного результата исполнения запросов и др. [9].

Случайный сэмплинг является одним из первых предложенных и хорошо разработанных подходов к решению проблемы обработки сверхбольших баз данных [20]. В работе [7] представлен алгоритм случайного сэмплинга, позволяющий сохранить в сэмпле ограничения целостности оригинальной базы данных: ссылочная целостность по внешним ключам, функциональные зависимости, ограничения целостности доменов.

В настоящее время сэмплинг поддерживается в ряде СУБД корпоративного класса. В СУБД Oracle поддерживается выборка данных из случайного сэмпла указанной таблицы путем добавления в запрос SELECT специального ключевого слова SAMPLE [21]. СУБД IBM DB2 предоставляет два оператора, обеспечивающих сэмплинг, – RAND и TABLESAMPLE [14]. Функция RAND(), указанная в части WHERE запроса, позволяет задает долю кортежей оригинальной таблицы, случайно попадающих в сэмпл. Оператор TABLESAMPLE позволяет отбирать в кортежи в сэмпл одним из двух способов: на основе алгоритма, предложенного в работе [12], или случайным отбором физических страниц диска, на которых размещены кортежи.

В области интеллектуального анализа данных сэмплинг является одним из наиболее часто используемых методов для достижения баланса между вычислительной сложностью аналитической обработки большого массива данных и точностью получаемых результатов. Однако соответствующие алгоритмы сэмплинга, как правило, ориентированы на конкретные алгоритмы интеллектуального анализа, которые будут использовать сэмпл в качестве входных данных, и могут применяться только для реляционных баз данных, состоящих из единственного отношения [10, 22, 24]. В работе [18] предложен статический метод сэмплинга, который использует распределение данных сэмпла как критерий принятия решения, отражает ли сэмпл оригинальное множество данных. Данный подход, однако, ограничен применением для случая базы данных, состоящей из одного отношения. В работе [26] представлен алгоритм сэмплинга для реляционных баз данных, направленный на улучшение масштабируемости и точности методов классификации для мульти-реляционных данных.

Большое количество работ посвящено использованию случайного сэмплинга для быстрого нахождения приблизительного результата запроса [6, 13] и размера результирующего отношения [11]. В работе [16] предложен метод Linked Bernoulli Synopses, основанный на алгоритме Join Synopses (JS) [5] для вычисления приблизительного результата выполнения запроса, предполагающего соединение большого количества отношений. Оба подхода требуют обработки каждого кортежа базы данных. В случае JS каждый кортеж базы данных попадает в сэмпл с вероятностью, равной коэффициенту сэмплинга (отношению количества кортежей в сэмпле к количеству кортежей в оригинальной базе данных). После вставки кортежей в сэмпл JS гарантирует сохранение ссылочной целостности в сэмпле, дополнительно выполняя последовательный перебор всех отношений сэмпла, начиная с некоторого стартового отношения, и добавляя в каждое отношение пропущенные ссылаемые кортежи. Алгоритм LBS предполагает однократный запуск над всей базой данных. Для каждого кортежа в каждом отношении оригинальной базы данных вычисляется вероятность вставки данного кортежа в сэмпл, которая зависит от вероятностей вставки в сэмпл кортежей, ссылающихся на этот кортеж.

Алгоритм VFDS (Very Fast Database Sampling) [8] является одним из наиболее быстрых алгоритмов случайного сэмплинга баз данных. VFDS требует одного прохода над всей оригинальной базой данных и не выполняет обработку каждого ее кортежа в отдельности. На первом шаге случайным образом отбираются кортежи стартового отношения и вставляются в соответствующее отношение сэмпла. После этого алгоритм обеспечивает целостность данных в сэмпле посредством сэмплинга ссылающихся и ссылаемых кортежей стартового отношения. Процесс рекурсивно продолжается для ссылающихся и ссылаемых кортежей, попавших в сэмпл на предыдущем шаге. Сэмплинг завершается, как только обработаны все отношения оригинальной базы данных.

Репрезентативность является важным свойством сэмплинга, которое обеспечивает меньшие время и вычислительную сложность обработки сэмпла по сравнению с обработкой оригинальной базы данных, одновременно сохраняя похожесть результатов обработки. Репрезентативный сэмпл реляционной базы данных определяется [9] как сэмпл, отбор данных в который позволяет сохранить распределение значений определенных атрибутов оригинальной базы данных. Такими атрибутами, прежде всего, являются внешние ключи, поскольку они реализуют логические связи между кортежами различных отношений.

Алгоритм CoDS (Chains of Dependencies-based sampling) [9] является на сегодня, по-видимому, самым быстрым алгоритмом репрезентативного сэмплинга. Ключевой идеей алгоритма является определение кортежей, которые должны быть отобраны из стартового отношения. После того, как эти кортежи будут вставлены в сэмпл базы данных, выполняется вставка в сэмпл ссылающихся и ссылаемых кортежей из оставшихся отношений оригинальной базы данных. Подобно алгоритму VFDS, процесс является рекурсивным и завершается, когда просмотрены все отношения. Отличие заключается в том, что кортежи стартового отношения отбираются в сэмпл таким образом, чтобы в итоге обеспечить необходимую репрезентативность.

Разработанный нами алгоритм репрезентативного сэмплинга для параллельных систем баз данных, представленный в разделе 2, является модернизированной версией алгоритма CoDS. В силу этого далее мы приводим краткое описание данного алгоритма. CoDS состоит из четырех следующих этапов: выбор стартового отношения, нахождение цепочек зависимостей и построение соответствующих диаграмм рассеивания, анализ диаграмм рассеивания и финальный отбор кортежей в сэмпл.

На этапе *выбора стартового отношения* в качестве стартового берется любое отношение базы данных, не имеющее внешних ключей. Если таких отношений несколько, в качестве стартового выбирается то из них, которое имеет максимальную мощность.

Цепочка зависимостей (chain of dependencies) представляет собой последовательность, в которой перечислены два или более отношений оригинальной базы данных. В этой последовательности в двух любых соседних членах одно из них ссылается на другое посредством внешнего ключа. На втором этапе осуществляется поиск всех цепочек зависимостей, начинающихся со стартового отношения. Затем для каждой найденной цепочки осуществляется построение диаграммы рассеивания. *Диаграмма рассеивания (scatter plot)* представляет собой гистограмму, по оси абсцисс которой откладывается количество кортежей стартового отношения, а по оси ординат – количество кортежей из конечного отношения из цепочки зависимостей, для которой строится диаграмма. Точка на этой гистограмме означает, что x кортежей из стартового отношения связаны с y кортежей из конечного отношения цепочки. Кортежи стартового отношения, связанные с данной точкой на диаграмме рассеивания, называются *точкой данных (data point)*.

На этапе *анализа диаграмм рассеивания* осуществляется отбор кортежей стартового отношения, которые будут включены в сэмпл. Основной задачей здесь является уменьшение размеров каждой точки данных в соответствии с заданным коэффициентом сэмплинга. Для этого авторами алгоритма предложена *функция влияния (impact factor, IF)*, которая для заданного кортежа вычисляет степень нарушения репрезентативности сэмпла в случае попадания данного кортежа в сэмпл. Чем меньше значение функции влияния, тем больше вероятность включения в сэмпл данного кортежа.

На этапе *финального заполнения сэмпла* в отношения базы данных добавляются кортежи, которые связаны с кортежами, отобранными в стартовое отношение посредством внешних ключей.

В следующем разделе мы покажем, каким образом можно модернизировать алгоритм CoDS для параллельных реляционных систем баз данных на основе фрагментного параллелизма.

2. Алгоритм pCoDS репрезентативного сэмплинга параллельных систем баз данных. Рассмотренные в предыдущем разделе методы сэмплинга не могут быть применены для параллельных систем баз данных на основе фрагментного параллелизма, поскольку не учитывают особенности распределения фрагментов отношений базы данных по вычислительным узлам кластерной системы. Далее мы рассмотрим разработанную нами модернизированную версию алгоритма CoDS, названную нами *pCoDS*, В pCoDS при построении сэмпла делается попытка сохранить в нем следующие важные особенности оригинальной базы данных: соотношение размеров фрагментов отношений и соотношение «своих» и «чужих» кортежей в отношениях. «Своими» называют такие кортежи, которые при выполнении запроса должны быть обработаны на текущем вычислительном узле кластера, «чужими» – те, что должны быть переданы для обработки на другой вычислительный узел.

Сохранение относительных размеров фрагментов отношений направлено на обеспечение репрезентативности *перекоса по данным*, когда значение некоторые значения для определенного атрибута встречаются значительно чаще, чем остальные. Перекос по данным может привести к дисбалансу загрузки узлов кластера и катастрофически ограничить ускорение параллельной СУБД [19]. Поскольку фрагментация часто осуществляется на основе значений первичного ключа отношения, сохранение относительных размеров фрагментов отношений обеспечивает репрезентативность возможного дисбаланса загрузки вычислительных узлов кластера при выполнении запроса. Сохранение соотношения «своих» и «чужих» кортежей обеспечивает репрезентативность нагрузки, связанной с пересылкой данных между вычислительными узлами кластера при выполнении запроса.

В алгоритме pCoDS нами модифицируются следующие шаги исходного алгоритма: выбор стартового отношения, создание диаграмм рассеивания и анализ диаграмм рассеивания.

На этапе *выбора стартового отношения* в алгоритме CoDS выбирается отношение, не содержащее внешних ключей. Предложенная нами модификация алгоритма подразумевает, что в качестве стартового выбирается отношение, которое наиболее часто фигурирует в запросах к базе данных. Данная модификация направлена на обеспечение репрезентативности соотношения «своих» и «чужих» кортежей для запросов, содержащих в соединениях стартовое отношение.

На этапе *создания диаграмм рассеивания* мы вносим изменения в принцип построения диаграмм рассеивания. Для каждой цепочки зависимостей в pCoDS строится *две* диаграммы рассеивания: для «своих» кортежей (обработка которых осуществляется на текущем узле) и для «чужих» кортежей (обработка которых требует их пересылки на другой узел кластера) стартового отношения. В силу того, что размер всех точек данных из каждой диаграммы рассеивания уменьшается пропорционально коэффициенту сэмплинга, данная модификация позволяет сохранить соотношение «своих» и «чужих» кортежей для стартового отношения, выбранного в качестве стартового.

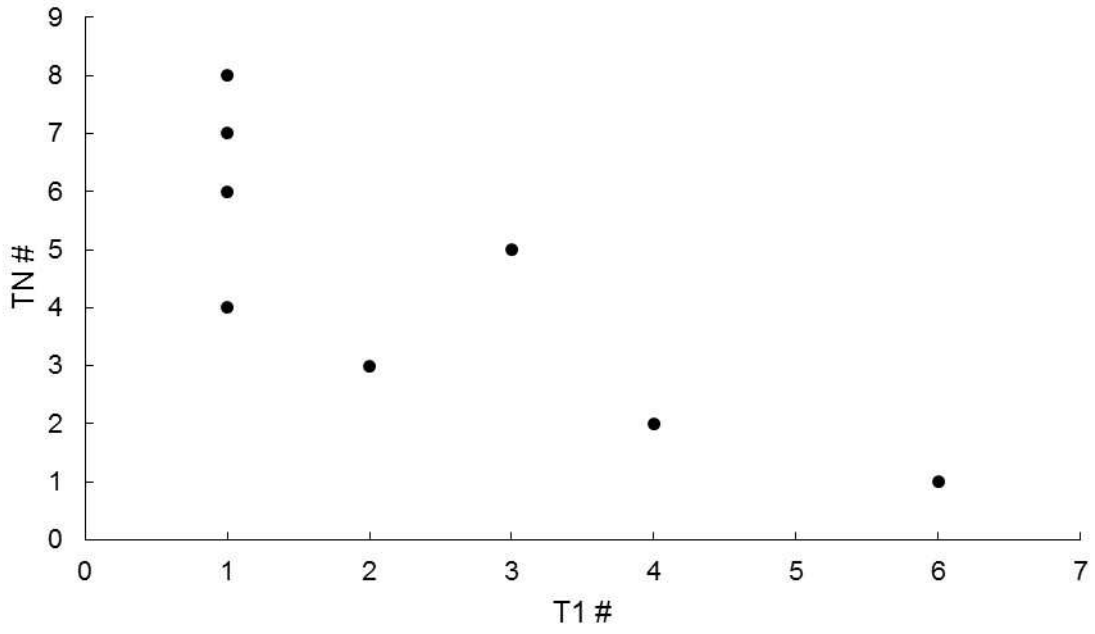

Рис. 1. Диаграмма рассеивания в CoDS

Пример диаграммы рассеивания для цепочки (T_1, T_2, T_3, …, T_N) в алгоритме CoDS приведен на рис. 1. Диаграммы рассеивания для той же цепочки в алгоритме pCoDS приведены на рис. 2, здесь φ означает функцию фрагментации отношения.

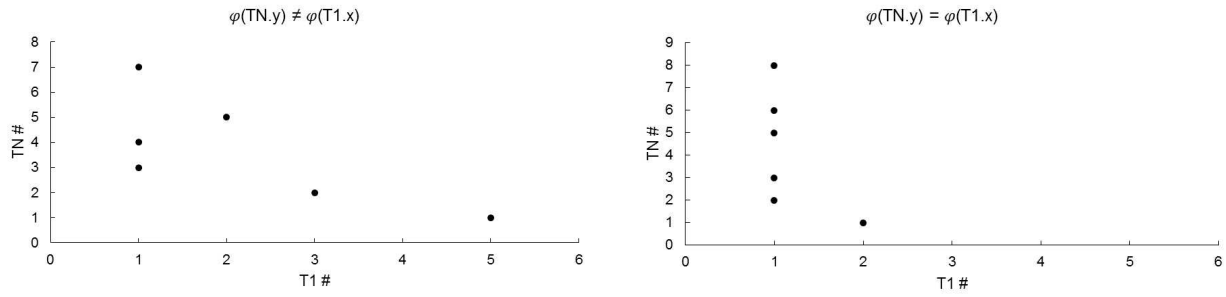

Рис. 2. Диаграммы рассеивания в алгоритме pCoDS

Запросы, используемые для создания диаграмм рассеивания в алгоритме pCoDS, приведены на рис. 3.

```
SELECT T1.ID, Count(DISTINCT TN.ID) as Y
FROM T1 JOIN T2 JOIN T3 JOIN … JOIN TN
WHERE φ(T1.ID) = φ(TN.ID)
GROUP BY T1.ID

SELECT T1.ID, Count(DISTINCT TN.ID) as Y
FROM T1 JOIN T2 JOIN T3 JOIN … JOIN TN
WHERE φ(T1.ID) <> φ(TN.ID)
GROUP BY T1.ID
```

Рис. 3. Запросы для создания диаграмм рассеивания в pCoDS

На этапе *анализа диаграмм рассеивания* мы изменяем функцию влияния, добавляя в нее дополнительное слагаемое с целью обеспечить репрезентативность относительных размеров фрагментов базы данных, и функция влияния принимает вид (1). В силу того, что функция влияния рассчитывается для каждого кортежа на основе кортежей, уже отобранных в сэмпл, предложенная модификация позволяет сохранить данную характеристику. С помощью коэффициента k регулируется степень важности сохранения данной характеристики.

$$IF(T^*.t) = IF_{CoDS}(T^*.t) + k * \frac{\|TS^*_{\varphi(t)}\| / \|TS^*\|}{\|T^*_{\varphi(t)}\| / \|T^*\|} \quad (1)$$

где

TS^* – стартовое отношение в сэмпле базы данных,

T^* – стартовое отношение в исходной базе данных,

φ – функция фрагментации стартового отношения,

t – обрабатываемый кортеж,

$k \in [0, 1]$– коэффициент, определяющий степень важности сохранения относительных размеров фрагментов,

$IFCoDS(T^*.t)$ – функция влияния исходного алгоритма CoDS, вычисляемая по формуле (2).

$$IF_{CoDS}(T^*.t) = \sum_{dp' \in RDP(dp)} \frac{\|dp' \cap TS^*\|}{\alpha * \|dp'\|} \quad (2)$$

где

dp – точка данных на диаграмме рассеивания,

$RDP(dp)$ – множество точек данных, содержащих одинаковые кортежи (связанные точки данных, related data points),

α – коэффициент сэмплинга.

3. Вычислительные эксперименты. Предложенный нами алгоритм был реализован для параллельной СУБД PargreSQL [2, 23]. Для исследования свойств реализованного алгоритма нами проведена серия вычислительных экспериментов, в которых использовалась финансовая база данных [15], предлагавшаяся участникам конкурса PKDD'99 Challenge Discovery, и использованная в экспериментах разработчиками алгоритма CoDS.

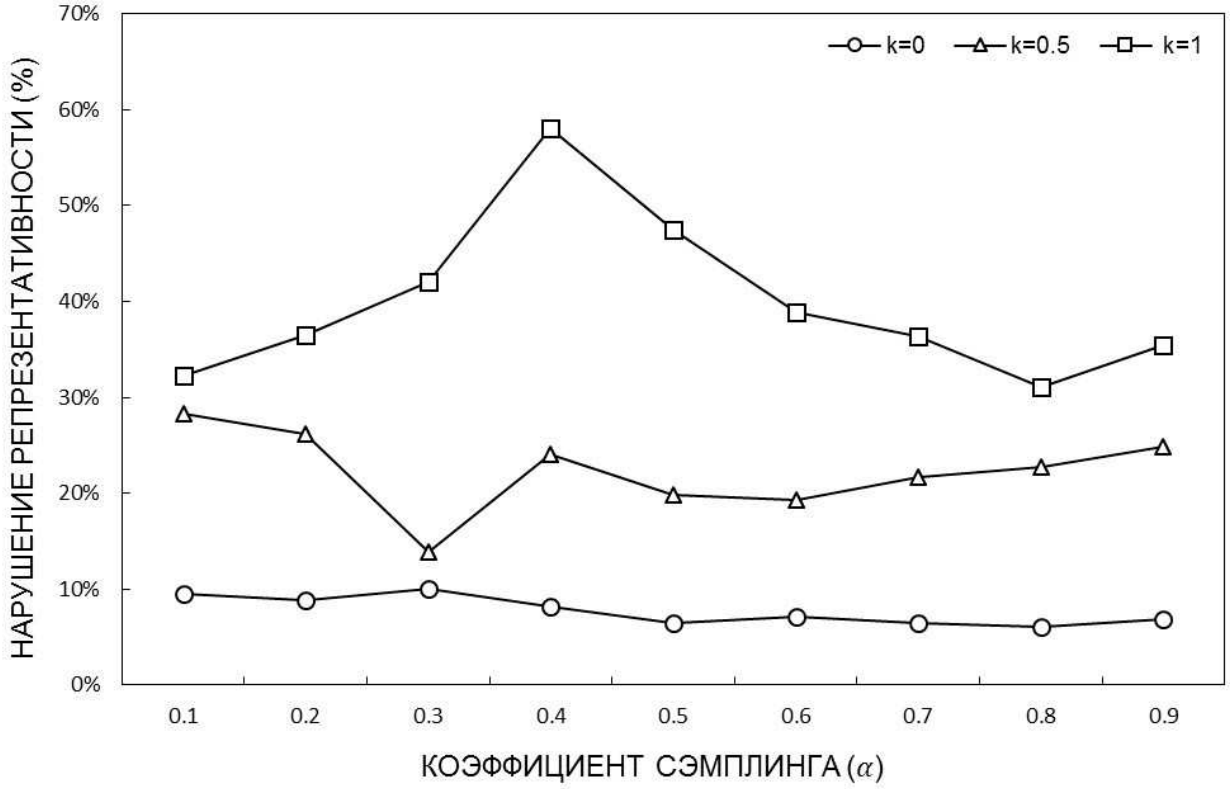

Рис. 4. Репрезентативность сэмпла

В рамках экспериментов исследовались следующие характеристики предложенного алгоритма: репрезентативность сэмпла, отклонение размера результирующего сэмпла, репрезентативность загрузки узлов кластера и репрезентативность нагрузки по пересылке кортежей между узлами при обработке запросов.

Репрезентативность базы данных. На рис. 4 приведены результаты экспериментов по измерению репрезентативности сэмпла. Можно видеть, что при увеличении коэффициента важности сохранения относительных размеров фрагментов, репрезентативность сэмпла ухудшается.

Для измерения использовалась мера (3), предложенная авторами алгоритма CoDS. Результирующее значение, равное нулю, будет означать идеальную репрезентативность сэмпла.

$$\delta_{\varphi}(S(T)) = \frac{1}{n}\sum_{k=1}^{n}\left|\frac{\|S(T)_k\|}{\|S(T)\|} - \frac{\|O(T)_k\|}{\|O(T)\|}\right| \quad (3)$$

где

t', t – отношения базы данных,

$\|sptt'\|$ – количество точек данных на диаграмме рассеивания между t' и t,

dptt' – точка данных из диаграммы рассеивания sptt',

S(T) – сэмпл базы данных,

$C = \|dptt' \cap S(T)\|$ – количество кортежей в точке данных в сэмпле,

$E = \alpha * \|dptt'\|$ – ожидаемое количество кортеже в точке данных в сэмпле,

RT(t) – отношения, связанные с t.

Отклонение размера сэмпла от заданного. На рис. 5 приведены результаты экспериментов по измерению отклонения размера сэмпла от заданного. Можно видеть, что увеличение коэффициента сохранения относительных размеров фрагмент не влияет на получаемый размер сэмпла. При этом сэмпл получается меньше, чем задаваемое значение в среднем на 18%, что связано с особенностями алгоритма CoDS и является ожидаемым результатом.

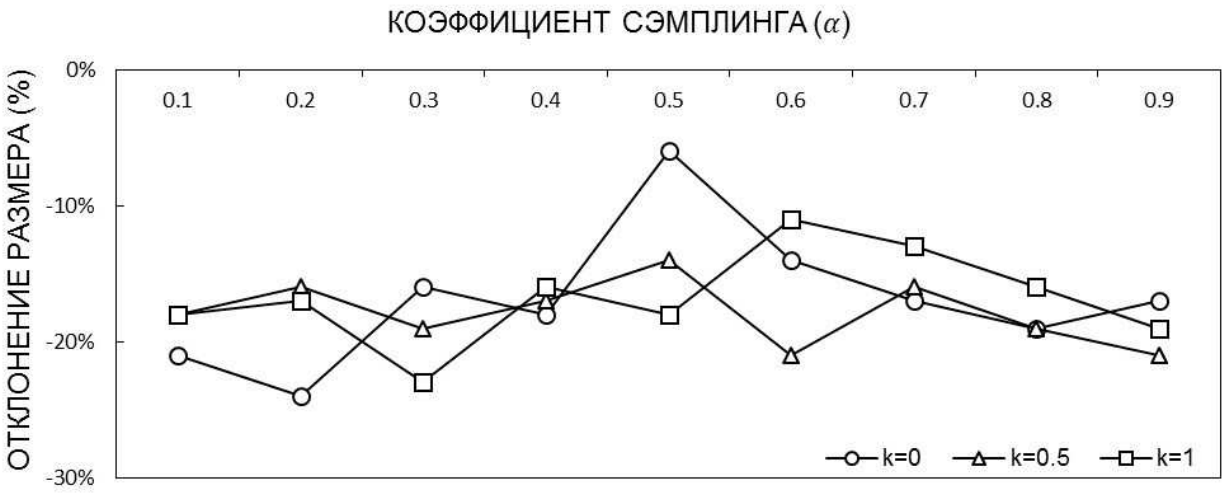

Рис. 5. Отклонение размера результирующего сэмпла от заданного

Для измерения соответствия размера сэмпла заданному использовалась мера (4), предложенная авторами алгоритма CoDS. Результирующее значение, равное нулю, будет означать идеальное соответствие размера сэмпла заданному.

$$\delta(S(T)) = \frac{1}{\|S(T)\|}\sum_{t\in T}\left(\frac{1}{\|RT(t)\|}\sum_{t'\in RT(t)}\left(\frac{1}{\|sp_t^{t'}\|}\sum_{dp_t^{t'}\in sp_t^{t'}}\min\left(\frac{|C-\lfloor E\rfloor|}{\lfloor E\rfloor},\frac{|C-\lceil E\rceil|}{\lceil E\rceil}\right)\right)\right) \quad (4)$$

Репрезентативность загрузки узлов кластера. Для измерения сохранения относительных размеров фрагментов нами предлагается мера (5). Результирующее значение равное нулю будет означать идеальную репрезентативность загрузки узлов базы данных.

$$\delta_{\|T\|}(S(T)) = \frac{\|S(T)\| - \alpha * \|O(T)\|}{\alpha * \|O(T)\|} \quad (5)$$

где
n – количество фрагментов,
S(T) – сэмпл базы данных,
O(T) – исходная база данных.

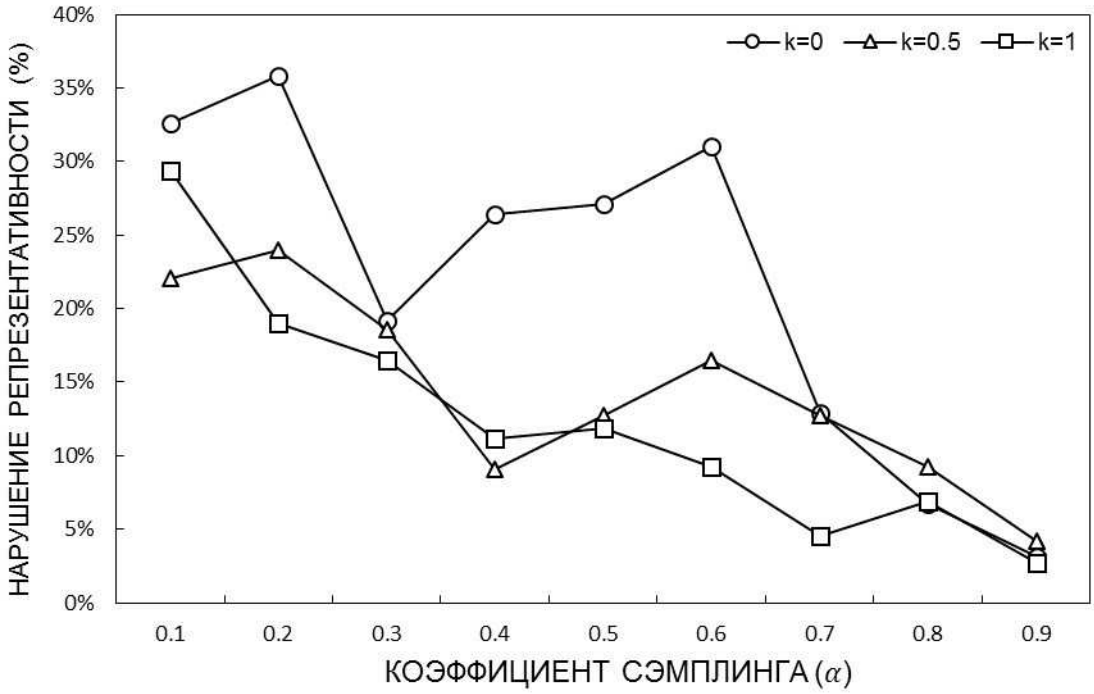

Рис. 6. Репрезентативность загрузки узлов кластера

На рис. 6 приведены результаты экспериментов по измерению репрезентативности загрузки узлов кластера. При увеличении коэффициента важности сохранения относительных размеров фрагментов, степень сохранения относительных размеров фрагментов (5) улучшается. При сравнении результатов при k=0 и k=1 среднее значение меры уменьшается почти вдвое с 0.236 до 0.124. Это позволяет заключить, что модификация функции влияния до вида (1) действительно улучшает репрезентативность дисбаланса загрузки узлов кластера.

Репрезентативность нагрузки по пересылке данных между узлами. Для измерения сохранения соотношения «своих» и «чужих» кортежей нами предлагается мера (6). Измерение проводилось на двух типах запросов: содержащих и не содержащих стартовое отношение в соединении.

$$\delta_{io} = \left| \frac{native_{S(T)}}{alien_{S(T)}} - \frac{native_{O(T)}}{alien_{O(T)}} \right| \quad (6)$$

где

nativeX – количество «своих» кортежей при запросе к базе данных X,

alienx – количество «чужих» кортежей при запросе к базе данных X.

Результаты экспериментов по измерению репрезентативности возможной нагрузки по пересылке данных между узлами приведены на рис. 7. Можно видеть, что модификация шага алгоритма, связанного с построением диаграмм рассеивания, позволяет сильно улучшить репрезентативность данной характеристики для запросов, содержащих в соединениях стартовое отношение. При этом для запросов, не содержащих стартовое отношение, степень репрезентативности также улучшается. Результаты показывают, что для запросов, содержащих стартовое отношение в соединениях, среднее значение меры уменьшается с 0.43 до 0.15, что является отличным результатом. При этом для запросов, не содержащих стартового отношения, среднее значение уменьшается с 0.44 до 0.36, что также является хорошим результатом.

Таким образом, результаты проведенных экспериментов свидетельствуют о том, что предложенный алгоритм сэмплинга pCoDS адекватно сохраняет важные характеристики параллельной системы базы данных и обеспечивает приемлемую репрезентативность данных.

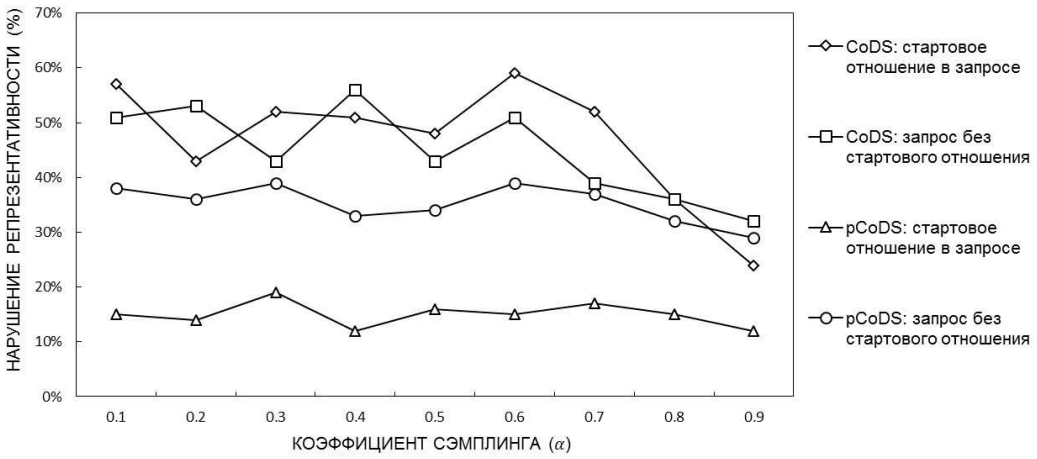

Рис. 7. Репрезентативность возможной нагрузки по пересылке данных между узлами кластера

Заключение. В статье рассмотрена разработка алгоритма репрезентативного сэмплинга для параллельных систем баз данных на основе фрагментного параллелизма. Предложена модификация существующего алгоритма CoDS для последовательных систем баз данных, который является одним из наиболее быстрых алгоритмов сэмплинга и показывает при этом приемлемую репрезентативность. Модифицированный алгоритм позволяет сохранять при сэмплинге следующие важные для параллельных СУБД характеристики распределения базы данных по вычислительным узлам кластерной системы: относительные размеры фрагментов отношений и относительное количество «своих» и «чужих» кортежей в каждом фрагменте отношения. Сохранение относительных размеров фрагментов обеспечивает репрезентативность возможного дисбаланса загрузки вычислительных узлов кластера при обработке запроса. Сохранение относительного количества «своих» и «чужих» кортежей (т.е. кортежей, обрабатываемых соответственно на текущем узле или пересылаемых на другой узел кластера) в фрагментах обеспечивает репрезентативность нагрузки, связанной с пересылкой данных между узлами кластера при выполнении запросов. Разработанный алгоритм реализован для параллельной СУБД PargreSQL. Представлены результаты вычислительных экспериментов на реальных данных, подтверждающие приемлемую репрезентативность предложенного алгоритма.

В качестве возможных направлений дальнейших исследований интересными представляются следующие задачи:

1) встраивание предложенного алгоритма сэмплинга в ядро параллельной СУБД PargreSQL для расширения синтаксиса запросов SELECT;

2) разработка меры репрезентативности базы данных, фрагментированной по узлам кластерной системы, и ее сэмпла, которая учитывала бы как статистические характеристики базы данных, так и рассмотренные в данной статье характеристики, важные для параллельной системы баз данных;

3) адаптация предложенного алгоритма сэмплинга для параллельных систем баз данных, поддерживающих частичную репликацию данных [1].

Исследование выполнено при финансовой поддержке Российского фонда фундаментальных исследований в рамках научного проекта № 12-07-00443-а.

ЛИТЕРАТУРА:

1. Костенецкий П.С., Лепихов А.В., Соколинский Л.Б. Технологии параллельных систем баз данных для иерархических многопроцессорных сред // Автоматика и телемеханика. 2007. № 5. С. 112-125.
2. Пан К.С., Цымблер М.Л. Разработка параллельной СУБД на основе последовательной СУБД PostgreSQL с открытым исходным кодом // Вестник ЮУрГУ. Серия «Математическое моделирование и программирование». 2012. № 18(277). Вып. 12. С. 112-120.
3. Соколинский Л.Б. Обзор архитектур параллельных систем баз данных // Программирование. 2004. № 6. С. 49-63.
4. Янцен Д.Д., Цымблер М.Л. Алгоритм репрезентативного сэмплинга для параллельных систем баз данных // Параллельные вычислительные технологии (ПаВТ'2014): труды международной научной конференции (1-3 апреля 2014 г., г. Ростов-на-Дону). Челябинск: Издательский центр ЮУрГУ, 2014. С. 381.
5. Acharya S., Gibbons P.B., Poosala V., Ramaswamy S. Join Synopses for Approximate Query Answering // SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA. ACM, 1999. P. 275-286.
6. Agarwal S., Panda A., Mozafari B., Iyer A.P., Madden S., Stoica I. Blink and It's Done: Interactive Queries on Very Large Data // Proceedings of the VLDB Endowment. 2011. Vol. 5, No. 1. P. 1902-1905.
7. Bisbal J., Grimson J., Bell D.A. A Formal Framework for Database Sampling // Information & Software Technology. 2005. Vol. 47, No. 1. P. 819-828.

8. Buda T.S. Generation of Test Databases using Sampling Methods // International Symposium on Software Testing and Analysis, ISSTA'13, Lugano, Switzerland, July 15-20, 2013. ACM, 2013. P. 366-369.
9. Buda T.S., Cerqueus T., Murphy J., Kristiansen M. CoDS: A Representative Sampling Method for Relational Databases // Database and Expert Systems Applications – 24th International Conference, DEXA 2013, Prague, Czech Republic, August 26–29, 2013. Proceedings, Part I. Springer, 2013. Lecture Notes in Computer Science. Vol. 8055. P. 342-356.
10. Chakaravarthy V.T., Pandit V., Sabharwal Y. Analysis of sampling techniques for association rule mining // Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings. ACM, 2009. P. 276-283.
11. Chaudhuri S., Das G., Srivastava U. Effective Use of Block-Level Sampling in Statistics Estimation // Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004. ACM, 2004. P. 287-298.
12. Gemulla R., Rösch P., Lehner W. Linked Bernoulli Synopses: Sampling along Foreign Keys // Scientific and Statistical Database Management, 20th International Conference, SSDBM 2008, Hong Kong, China, July 9-11, 2008, Proceedings. Springer, 2008. Lecture Notes in Computer Science. P. 6-23.
13. Goethals B., Page W.L., Mampaey M. Mining Interesting Sets and Rules in Relational Databases // Proceedings of the 2010 ACM Symposium on Applied Computing (SAC), Sierre, Switzerland, March 22-26, 2010. ACM, 2010. P. 997-1001.
14. Gryz J., Guo J., Liu L., Zuzarte C. Query Sampling in DB2 Universal Database // Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004. ACM, 2004. P. 839-843.
15. Guide to the Financial Data Set of the PKDD'99 Discovery Challenge. URL: http://lisp.vse.cz/pkdd99/Challenge/berka.htm (дата обращения: 24.05.2014).
16. Haas P.J., Koenig C. A Bi-Level Bernoulli Scheme for Database Sampling // Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004. ACM, 2004. P. 275-286.
17. Ioannidis Y.E., Poosala V. Histogram-Based Approximation of Set-Valued Query-Answer // VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK. Morgan Kaufmann 1999. P. 174-185.
18. John G.H., Langley P. Static Versus Dynamic Sampling for Data Mining // Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA. AAAI Press, 1996. P. 367-370.
19. Lakshmi M.S., Yu P.S. Effectiveness of Parallel Joins // IEEE Transactions on Knowledge and Data Engineering. 1990. Vol. 2, No. 4. P. 410-424.
20. Olken F., Rotem D. Random Sampling from Database Files: A Survey // Statistical and Scientific Database Management, 5th International Conference SSDBM, Charlotte, NC, USA, April 3-5, 1990, Proceedings. Springer, 1990. Lecture Notes in Computer Science. P. 92-111.
21. Oracle Documentation. URL: http://docs.oracle.com/cd/E11882_01/server.112/e41084/statements_10002.htm (дата обращения: 24.05.2014).
22. Palmer C.R., Faloutsos C. Density Biased Sampling: an Improved Method for Data mining and Clustering // Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA. ACM, 2000. P. 82-92.
23. Pan C.S., Zymbler M.L. Taming Elephants, or How to Embed Parallelism into PostgreSQL // Database and Expert Systems Applications - 24th International Conference, DEXA 2013, Prague, Czech Republic, August 26-29, 2013. Proceedings, Part I. Springer, 2013. Lecture Notes in Computer Science. Vol. 8055. P. 153-164.
24. Parthasarathy S. Efficient Progressive Sampling for Association Rules // Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan. IEEE, 2002. P. 354-361.
25. Wu X., Wang Y., Guo S., Zheng Y. Privacy Preserving Database Generation for Database Application Testing // Fundamenta Informaticae. 2007. Vol. 78, No. 1. P. 595-612.
26. Yin X., Han J., Yang J., Yu P.S. Efficient Classification across Multiple Database Relations: A CrossMine Approach // IEEE Transactions on Knowledge and Data Engineering. 2006. Vol. 18, No. 1. P. 770-783.